## **Shopping Cart Services**

### **Guidelines -**

- Follow all standard coding practices.
- Design efficient database tables for storing data.
- Use Test Driven Development (TDD)(JUnits for backend).
- Follow proper error handling.
- Write comments in code.
- Impose necessary validations at required places(mobile, e-mail etc.).
- UI should have a professional look and feel.
- UI code should work in all browsers(Cross browser Compatibility)
- Use Angular, Java (11) and mysql.

### Requirements -

### > Login Management :

- User login/logout.
- forgot password.
- New user signup.
- User Profile to update user details

### ➤ Home Page & Filters :

Landing page, categories view of procuts with working filters, show products on a set of filters. Filters functionalities at least should cover price, brand, category along with sorting.

### > Product Management :

- single product view with 'add to cart' feature.
- Add/Edit Products by Admin
- Paginated Products.

### Cart Management:

Cart should support at least 1 product with varied quantity per product

- delete from cart.
- empty cart.
- checkout.
- edit cart
- order history.

Please find Sample APIs for above functionalities.

1. Login
URI : /login
HTTP VERB : POST
REQUEST : {     "email":"xyz@beehyv.com",     "password":" <base 64="" encoded="" string=""/> " }
RESPONSE : {"result":"Success"} 200 Ok {"result":"failure"} 401
2. Signup

2.	Signup		
URI:	URI : /signup		
НТТР	VERB : POST		

REQUEST: {     "name":"xyz",     "email":"xyz@beehyv.com",     "password":" <base 64="" encoded="" string=""/> " }
RESPONSE : {"userId":" <user number="" sequence="">"} 200 Ok</user>
3. Logout
URI : /logout
HTTP VERB : POST
REQUEST : { "userID":"234" }
RESPONSE : {"result":"Success"} 200 Ok {"result":"failure"} 401

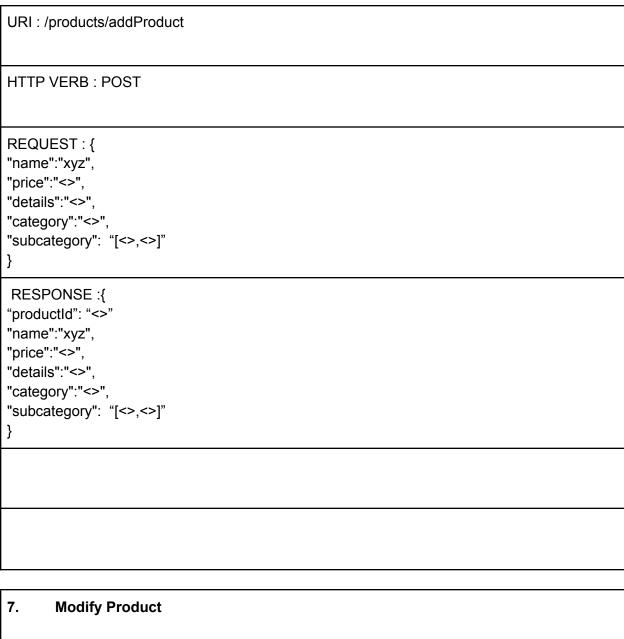


```
4.
       Get Profile
URI : /getprofile/{userId}
HTTP VERB: GET
REQUEST: {}
RESPONSE: {
"userID":"<ID of User>",
"name":"xyz",
"email":"xyz@beehyv.com",
"phone":"<mobile>",
"address": {
       "street": "<>",
       "city": "<>",
       "state": "<>",
       "pincode" : "<>"
}
```



5. Update Profile
URI : /updateProfile
HTTP VERB : POST
REQUEST: {  "userID":" <id of="" user="">",  "name":"xyz",  "email":"xyz@beehyv.com",  "phone":"<mobile>",  "address": {      "street": "&lt;&gt;",      "city": "&lt;&gt;",      "state": "&lt;&gt;",      "pincode": "&lt;&gt;"  }</mobile></id>
RESPONSE :{"result":"Success"} 200 Ok {"result":"failure"} 401

### 6. Add Product



# 7. Modify Product URI: /products/update HTTP VERB: POST REQUEST: { "productId": "<>" "name":"xyz", "price":"<>",

```
"details":"<>",
    "category": "{<>,<,...]"
}

RESPONSE: {
    "productId": "<>"
    "name":"xyz",
    "price":"<>",
    "details":"<>",
    "category": "{<>,<,...]"
}

subcategory": "{<>,...]"
}
```

```
8. Get Product

URI: /products/getByld/{productId}

HTTP VERB: GET

REQUEST: {}

RESPONSE: {
  "productId": "<>"
  "name":"xyz",
  "price":"<>",
  "details":"<>",
  "category":"<>",
  "subcategory": "[<>,<>]"
}
```

9. Get Products by category
URI : /products/{category}
HTTP VERB : GET
REQUEST: {}
RESPONSE: Array of all{ "productId": "<>" "name":"xyz", "price":"<>", "details":"<>", "category":"<>", "subcategory": "[<>,<>]" }
10. Get Product by Search String
URI : /products/search/{searchString}

```
HTTP VERB : GET

REQUEST : {}

RESPONSE : Array of Searched {
  "productId": "<>"
  "name":"xyz",
  "price":"<>",
  "details":"<>",
  "category": "<>",
  "subcategory": "[<>,<>]"
}
```

# 11. Get Filtered Product by category

URI: /products/{category}/getFilteredProducts

HTTP VERB: POST

REQUEST: {["filterName": "filterValue", ...]}

```
RESPONSE: Array of filtered {

"productId": "<>"
"name":"xyz",

"price":"<>",

"details":"<>",

"category":"<>",

"subcategory": "[<>,<>]"
}
```

12. Get cart
URI : /cart/{userId}/getCart
HTTP VERB : GET
REQUEST:
RESPONSE : { cartId: <>, products: [ cartItem: <>] *cartItem given below
13. Get cart Item
URI : /cart/{userId}/getCartItem/{cartitemId}
HTTP VERB : GET
REQUEST:
RESPONSE : { cartItemId: <>, product : { "productId": "<>" "name":"xyz",

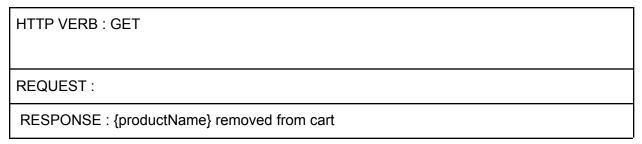
```
"price":"<>",
"details":"<>",
"category":"<>",
"subcategory": "[<>,<>]"
},
quantity: <Int>
```

```
14.
       Add To Cart
URI : /cart/{userId}/add/{productId}
HTTP VERB: GET
REQUEST:
RESPONSE: {
cartItemId: <>,
product : {
"productId": "<>"
"name":"xyz",
"price":"<>",
"details":"<>",
"category":"<>",
"subcategory": "[<>,<>]"
},
quantity: <Int>
```

### 15. Remove from Cart

URI : /cart/{userId}/remove/{productId}

\*if cart have already that product increase the quantity



# 16. Change Quantity of product in cart URI: /cart/{userId}/changeQuantity/{productId} || /cart/changeQuantity/{cartItemId} HTTP VERB: POST REQUEST: {quantity: <>} RESPONSE: { cartItemId: <>, product: { "productId": "<>" "name":"xyz", "price":"<>", "details":"<>", "category": "<>", "subcategory": "[<>,<>]"

quantity: <Int>

17.	Get order history
URI : /	order/{userId}/getOrders
HTTP	VERB : GET
REQU	EST:
RESF	ONSE : [{ d: <>,

products: [ orderItem: <>],	
producte: [ cracintein: ],	
orderStatus: <>	
}]	
11	
*orderItem similar to cartItem	

18. Create order	
URI : /order/{userId}/createOrder	
HTTP VERB : GET	
REQUEST:	
RESPONSE : { orderId: <>, products: [ orderItem: <>] orderStatus: <>} *orderItem given below	